Project1: Tweet Similarity

Documentation

# Design Process

Before starting coding we first search for the mathematical equation of the Cosine Similarity. Understand it, write down it on a white paper. To understand the behavior of the equation, we manually solve some of the examples and made sure for ourselves that the equation is correct and we are doing it in the right way.

After understanding the equation, we do some brainstorming on the algorithm. After some failures we made an algorithm and convert it in a simple C++ coding style.

The UML design which we used was the object oriented model. Because according to our understanding it meet our requirements and one of the easiest because of modularity and readability and also reusability of coding techniques.

# Data Structures

Reusability or modularity of code is not just give the justification of the program optimization. For optimization, we have to use some optimized and best time complexity data structures. In our code, we used vectors because in C++ vector is the only data structure against an array, even a dynamic array later required size but vector doesn’t need any size for creating or capturing memory during compile time.

For stop words we used data structure hash set, one of the best thing about this data structure is that it doesn’t store duplicate values. O(1) adds an element to this set, if it was not already there and O(1) returns true if the specified value is in this set.

# System Functionality

The functionality of system is very simple, when we run or execute the program it will ask for input text file means the file which contains training and testing tweets. And then it will ask for stopping words file.

The **ReadInputFile(inputfile)** method will separate testing and training tweets and store it in their variables.

**ReadStopWordsFile(stopfile)** method will read all the stop words and store it in its desire variable.

**FileWriter()** this function will be called and all the required methods one by one will be called and execute.

* **RemoveStopWords(testingTweets)** and **RemoveStopWords(rawTweets)** these two methods will remove all the stop words and then store it another variable.
* After this in two loops **CosineSimilarity(testingTweetsStoppedWordRemoved[i], tweetsStoppedWordRemoved[j])** this will return cosine similarity value of testing tweet against each training tweet.
* In cosine similarity method it first identify all the distinct words
* And prepare word frequencies for each tweet
* And finally the cosine similarity is obtained.

# Team Work

Kindly mention the team work by yourself.